11-29-00

A

[redacted]VLETT-PACKARD COMPANY
[redacted]tual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 10001161-1

11/28/00 U.S. PTO

JC846 U.S. PTO 09/724616 11/28/00

IN THE U.S. PATENT AND TRADEMARK OFFICE
Patent Application Transmittal Letter

COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Sir:

Transmitted herewith for filing under 37 CFR 1.53(b) is a(n): (X) Utility ( ) Design

(X) original patent application,

( ) continuation-in-part application

INVENTOR(S): **Michael L. Ziegler, Carol L. Thompson**

TITLE: **Method and Apparatus for Resuming Execution of a Failed Computer Program**

Enclosed are:

(X) The Declaration and Power of Attorney. (X) signed ( ) unsigned or partially signed

(X) 4 sheets of drawings (one set) ( ) Associate Power of Attorney

( ) Form PTO-1449 ( ) Information Disclosure Statement and Form PTO-1449

( ) Priority document(s) ( ) (Other) _____ (fee $_____)

CLAIMS AS FILED BY OTHER THAN A SMALL ENTITY				
(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) TOTALS
TOTAL CLAIMS	11 — 20	0	X $18	$ 0
INDEPENDENT CLAIMS	5 — 3	2	X $80	$ 160
ANY MULTIPLE DEPENDENT CLAIMS	0		$270	$ 0
BASIC FEE: Design ($320.00); Utility ($710.00)				$ 710
			TOTAL FILING FEE	$ 870
			OTHER FEES	$
			TOTAL CHARGES TO DEPOSIT ACCOUNT	$ 870

Charge $ 870 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16, 1.17,1.19, 1.20 and 1.21. A duplicate copy of this sheet is enclosed.

"Express Mail" label no. EL571600602US

Date of Deposit Nov. 28, 2000

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Commissioner for Patents, Washington, D.C. 20231.

By Keri J. Kuhlmann

Typed Name: **Keri J. Kuhlmann**

Respectfully submitted,

Michael L. Ziegler, Carol L. Thompson

By _____

LeRoy D. Maunu

Attorney/Agent for Applicant(s)
Reg. No. 35,274

Date: Nov. 28, 2000

Telephone No.: (651) 686-6633

Rev 10/00 (TransNew)

- Attach as First Page to Transmitted Papers -

METHOD AND APPARATUS FOR RESUMING EXECUTION OF A FAILED COMPUTER PROGRAM

FIELD OF THE INVENTION

The present invention generally relates to recovery from errors in computer programs, and more particularly to using alternative code to recover execution of a failed computer program.

BACKGROUND

Certain types of software errors are fatal to program execution. For example, a reference to a memory address that is beyond the address domain of a program will likely result in a fatal error. Certain timing or other transient conditions may also trigger fatal errors. While certain errors may be within the control of the software developer, the developer may be unable to guard against certain other errors in developing the software.

Program errors that are transient or timing-related are problematic to both the program user and to the program developer. From the user's point of view, failure of the program not only interrupts the task at hand, but may also result in lost work product. Reporting such transient errors is also difficult since the conditions required to reproduce the problem are likely to be unknown.

From the developer's point of view, much time may be spent trying to find the root cause of a problem where the root cause is external to the program. Furthermore, tracing the root cause of the problem may be difficult and time-consuming if there is there is scant information available for reproducing the problem.

A method and apparatus that address the aforementioned problems, as well as other related problems, are therefore desirable.

SUMMARY OF THE INVENTION

In various embodiments, a computer implemented method is provided for recovery from fatal program errors. A program is compiled using two compilers to generate first and second sets of object code. Checkpoints are identified in the program, and checkpoint code is generated for execution at the checkpoints. If execution of the first set of object code fails, checkpoint data is recovered and execution of the program is resumed using either the first or second set of object code. In one embodiment, the first set of object code is re-executed before trying the second set of object code. In another embodiment, the second set of object code automatically executed upon failure of the first set of object code.

It will be appreciated that various other embodiments are set forth in the Detailed Description and Claims which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

Various aspects and advantages of the invention will become apparent upon review of the following detailed description and upon reference to the drawings in which:

FIG. 1 is a flowchart of a an example process for compiling a program in accordance with one embodiment of the invention;

FIG. 2 is a block diagram that illustrates an example data structure used in the management of checkpoints;

FIG. 3 is a block diagram that illustrates the code that results from compilation of a program in accordance with one embodiment of the invention; and

FIG. 4 is a flowchart of an example process for error recovery.

DETAILED DESCRIPTION

In various embodiments, the present invention provides a method and apparatus for generating alternative code that supports recovery from a fatal program error.

FIG. 1 is a flowchart of a an example process for compiling a program in accordance with one embodiment of the invention. The process generally entails generating two sets of object code segments using two different compilers or code generators. The second set of object code segments are available for execution in the event that the a fatal program error is encountered in executing the first set of segments. The segments of object code are delineated by checkpoints that are identified by the compiler.

At steps 202 and 204, the program source code is compiled using two different compilers, and two sets of object code that are functionally equivalent are created from the compilation. For example, the compilers may be different versions of the same compiler or compilers from different vendors. For making the two sets of object code interchangeable, checkpoints are identified and checkpoint code is generated for the two sets of object code at step 206. Known computer platforms allow multiple compilers to coexist and also allow inter-operation at the function/procedure call level.

In one embodiment, the checkpoints are identified and the code generated as described in the co-pending patent application entitled, "Compiler-based Checkpointing for Support of Error Recovery" by Ziegler et al. and filed on October 31, 2000, which has patent/application number ***** and attorney docket number 10001159, is commonly assigned to the assignee of the present invention, and the contents of which are incorporated herein by reference. Each segment of code is delineated by a checkpoint, as determined by the compiler. The checkpoints are points in

the code where the state of the program is stored so that execution can be recovered at the point in the program following the checkpoint. For example, convenient places for checkpoints are procedure boundaries.

At step 208, the code that performs the checkpointing is generated and combined with the intermediate code that was generated from the user's source code. Along with the checkpointing code, a data structure is created for storage of the checkpoint data at step 208.

FIG. 2 is a block diagram that illustrates an example data structure used in the management of checkpoints in accordance with one embodiment of the invention. To save storage space, two checkpoint data sets 260a and 260b are maintained.

Checkpoint data are alternately stored in checkpoint data sets 260a and 260b for consecutive checkpoints. For example, at time t1, checkpoint data set 260a references checkpoint 262 and checkpoint data set 260b references checkpoint 264. At time t2 after program execution completes checkpoint 266, checkpoint data set 260a references checkpoint 266, and checkpoint data set 260b references checkpoint 264.

Timestamps or commit flags may be used in alternative embodiments to indicate which of the checkpoint data sets is to be used in recovery. The timestamp scheme involves writing a timestamp to a checkpoint data set when the storage of state information in the checkpoint data set is complete. Thus, the later of the two timestamps indicates which of checkpoint data sets 260a or 260b is to be used in recovery. The commit flag scheme involves a flag that indicates which of checkpoint data sets 260a or 206b is to be used in recovery.

FIG. 3 is a block diagram that illustrates the code that results from compilation of a program in accordance with one embodiment of the invention. One purpose for compiling code in the manner taught herein is to enable recovery from fatal program errors. Block 102 represents program source code that is to be compiled and is comprised of *n* segments of source code. Checkpoints are used to delineate the multiple segments. A checkpoint is a location in the code at which execution can recommence should the program encounter a fatal error. At each checkpoint, the state of data elements used by the program are stored so that in the event of program failure the state information can be recovered and execution resumed immediately after the checkpoint from which the state was recovered. In various embodiments the checkpoints can be user-programmed or identified by the compiler using recognized techniques.

Compilation of the program source code results in program object code 104 that includes two sets of object segments, object segments 1-*n* and object segments 1-*n'*. The object segments in each set correspond to the source segments of program source code 102.

Each set of object segments is code that is generated in compiling the source code with different compilers or code generators. In other words, object segments 1-*n* are generated by a first compiler, and object segments 1'-*n'* are generated by a different compiler. If the program fails during execution of segment *i*, for example, then the state of the checkpoint data can be recovered from checkpoint that precedes segment *i* and execution can resume at segment *i'*.

FIG. 4 is a flowchart of an example process for error recovery in accordance with one embodiment of the present invention. The process generally entails recovering from

a fatal program error by restoring checkpoint data and resuming execution of the program. In resuming execution, the code of the first set of segments is retried. If the failure is repeated, execution of the program is resumed using the alternative set of segments.

The process begins when a program error has been detected by the operating system, for example. At step 304, checkpoint data is restored, and at step 306 the program counter is reset to the selected checkpoint. The segment of object code from the first set of segments is re-executed at step 308. If the program executes the segment without error, decision step 310 and step 312 illustrate that the program continues with execution of the segments from the first set.

If an error recurs in executing the segment of code from the first set, control is directed to decision step 314, which determines whether the alternative code should be tried. In one embodiment, the first set of segments of code may be re-executed a selected number of times before trying execution of the alternative code. Control is directed to step 316 when the decision is made to execute the alternative code.

At step 316, checkpoint data is restored, and at step 318, the address of the segment of object code from the second set is selected for execution. That is, the program counter is loaded with a program address of a segment in the second set. At step 320, the alternative segment of object code is executed, and execution of the segments of code from the second set continues at step 322.

Returning now to decision step 314, if the decision is made to not execute the alternative code, control is directed to decision step 324. Decision step 324 tests whether execution of the segment from the first set of segments should be attempted again. If so, control is

returned to step 304 to restore the checkpoint data and try again. Otherwise, the program is exited with an error.

The present invention is believed to be applicable to compilers for a variety of programming languages. Other aspects and embodiments of the present invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and illustrated embodiments be considered as examples only, with a true scope and spirit of the invention being indicated by the following claims.

CLAIMS

What is claimed is:

1. A computer-implemented method for software error recovery, comprising:

compiling program source code into a first set of object code with a first compiler;

compiling the program source code into a second set of object code with a second compiler;

identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code;

associating sets of data objects with the checkpoints;

automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data objects for recovery if execution of the program is interrupted;

executing the first set of object code;

storing the state information in executing the checkpoint code; and

upon detecting an error in execution of the first set of object code, resuming execution of the program using the second set of object code.

2. The method of claim 1, further comprising:

upon detecting an error in execution of the first set of object code, initially re-executing the first set of object code; and

resuming execution using the second set of object code if the first set of object code fails in re-execution.

3. The method of claim 2, further comprising re-executing the first set of object code a selected number of times before resuming execution using the second set of object code.

4. The method of claim 3, further comprising ceasing resumption of execution of the first and second sets of object code if an error is detected in executing both sets of object code.

5. A computer-implemented method for software error recovery, comprising:

compiling program source code into a first set of object code with a first compiler;

compiling the program source code into a second set of object code with a second compiler;

identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code;

associating sets of data objects with the checkpoints;

automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data objects for recovery if execution of the program is interrupted;

executing the first set of object code;

storing the state information in executing the checkpoint code; and

upon detecting an error in execution of the first set of object code, selecting between the first set of object code and the second set of object code in resuming execution of the program.

6. The method of claim 5, further comprising:

upon detecting an error in execution of the first set of object code, initially re-executing the first set of object code; and

resuming execution using the second set of object code if the first set of object code fails in re-execution.

7. The method of claim 6, further comprising re-executing the first set of object code a selected number of times before resuming execution using the second set of object code.

8. The method of claim 7, further comprising ceasing resumption of execution of the first and second sets of object code if an error is detected in executing both sets of object code.

9. An apparatus for software error recovery, comprising:

means for compiling program source code into a first set of object code with a first compiler;

means for compiling the program source code into a second set of object code with a second compiler;

means for identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code;

means for associating sets of data objects with the checkpoints; and

means for automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data objects for recovery if execution of the program is interrupted;

means for executing the first set of object code;

means for storing the state information in executing the checkpoint code; and

means for resuming execution of the program using the second set of object code upon detecting an error in execution of the first set of object code.

10. A computer-implemented method for software error recovery, comprising:

means for compiling program source code into a first set of object code with a first compiler;

means for compiling the program source code into a second set of object code with a second compiler;

means for identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code;

means for associating sets of data objects with the checkpoints;

means for automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data objects for recovery if execution of the program is interrupted;

means for executing the first set of object code;

means for storing the state information in executing the checkpoint code; and

selecting between the first set of object code and the second set of object code in resuming execution of the program upon detecting an error in execution of the first set of object code.

11. A computer program product configured for causing a computer to perform the steps comprising:

compiling program source code into a first set of object code with a first compiler;

compiling the program source code into a second set of object code with a second compiler;

identifying checkpoints in the first and second sets of object code, each checkpoint in the first set of object code corresponding to a checkpoint in the second set of object code;

associating sets of data objects with the checkpoints;

automatically generating executable checkpoint code for execution at the checkpoints, the checkpoint code configured to store state information of the associated data objects for recovery if execution of the program is interrupted;

executing the first set of object code;

storing the state information in executing the checkpoint code; and

upon detecting an error in execution of the first set of object code, selecting between the first set of object code and the second set of object code in resuming execution of the program.

ABSTRACT

Method and apparatus for resuming execution of a failed computer program. A program is compiled using two compilers to generate first and second sets of object code. Checkpoints are identified in the program, and checkpoint code is generated for execution at the checkpoints. If execution of the first set of object code fails, checkpoint data is recovered and execution of the program is resumed using either the first or second set of object code. In one embodiment, the first set of object code is re-executed before trying the second set of object code.

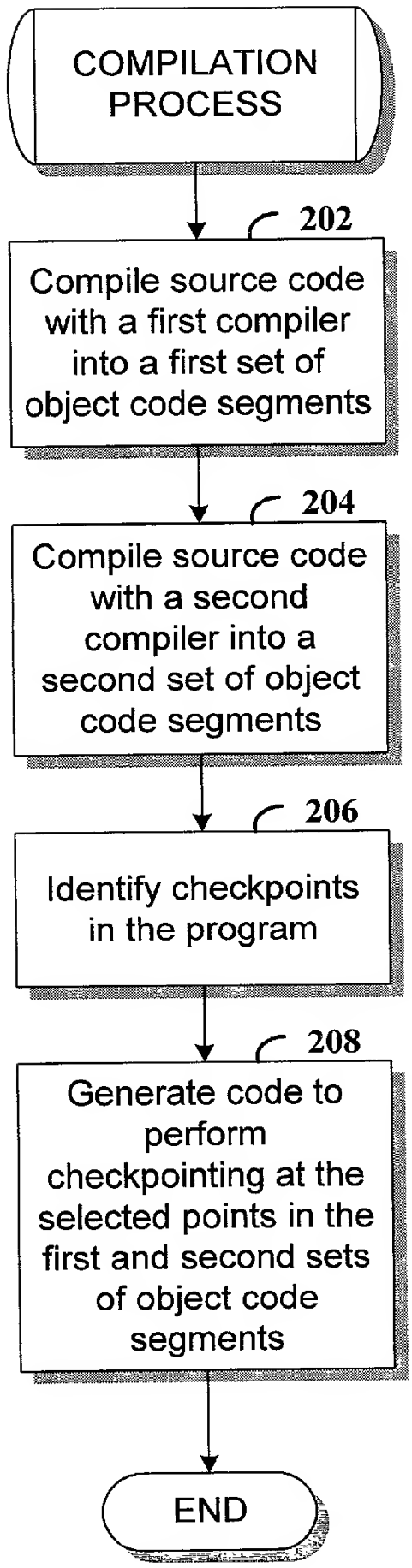
COMPILATION PROCESS
202
Compile source code with a first compiler into a first set of object code segments
204
Compile source code with a second compiler into a second set of object code segments
206
Identify checkpoints in the program
208
Generate code to perform checkpointing at the selected points in the first and second sets of object code segments
END

FIG. 1

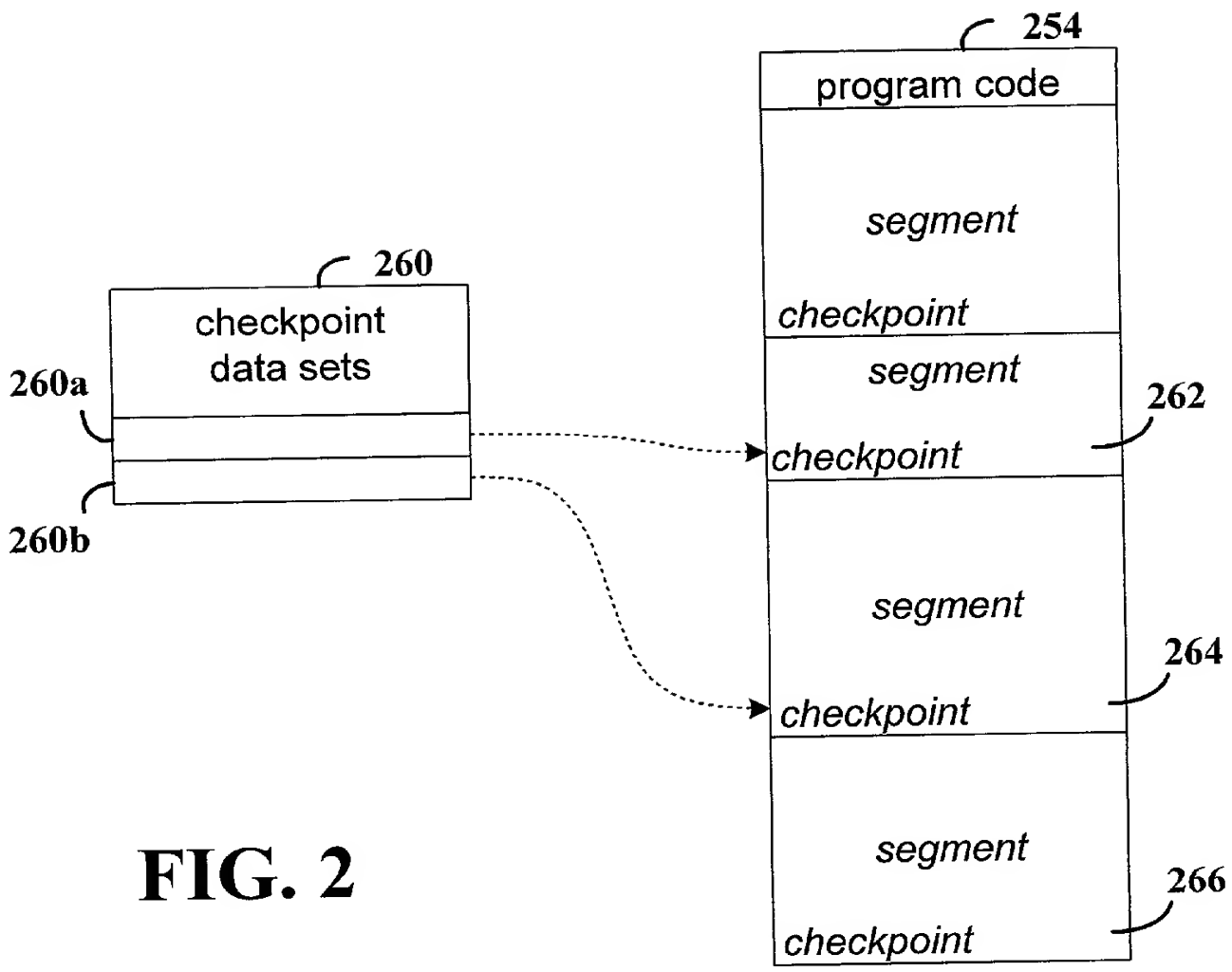
254
program code
segment
checkpoint
260
checkpoint
data sets
260a
segment
checkpoint
262
260b
segment
checkpoint
264
segment
checkpoint
266

FIG. 2

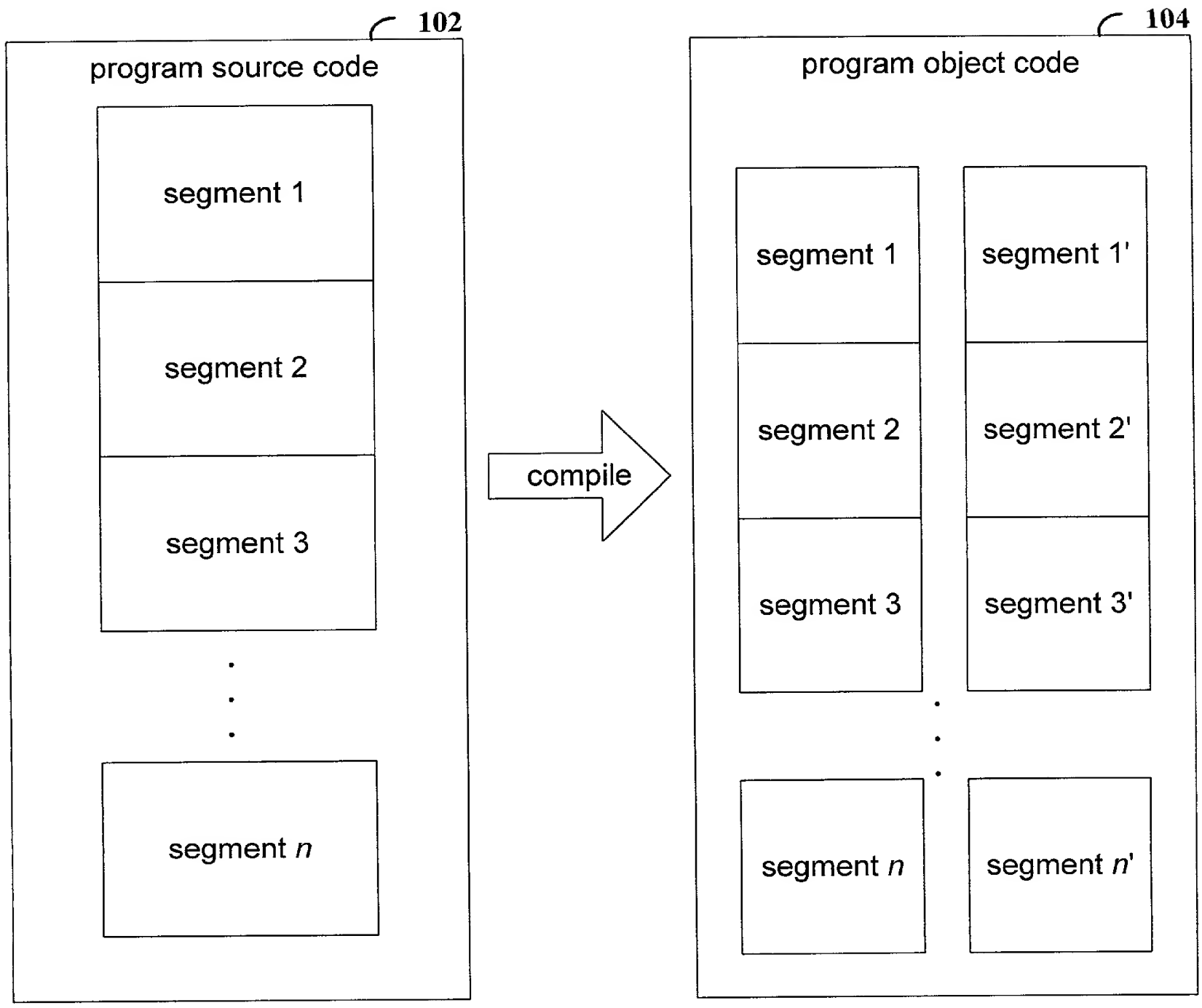
102
program source code
segment 1
segment 2
segment 3
segment n
compile
104
program object code
segment 1
segment 2
segment 3
segment n
segment 1'
segment 2'
segment 3'
segment n'

FIG. 3

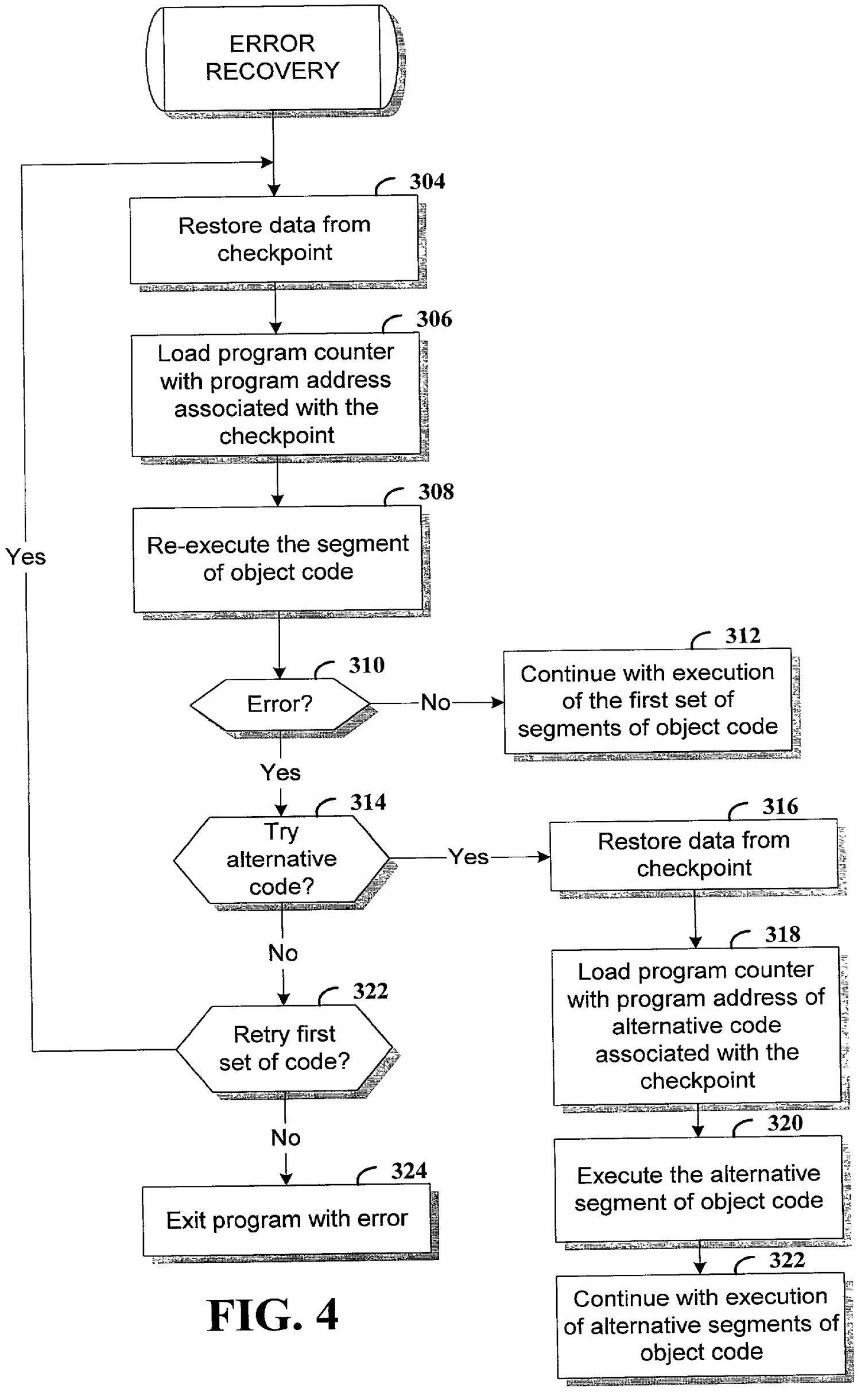
ERROR RECOVERY
304
Restore data from checkpoint
306
Load program counter with program address associated with the checkpoint
308
Re-execute the segment of object code
310
Error?
No
312
Continue with execution of the first set of segments of object code
Yes
314
Try alternative code?
Yes
316
Restore data from checkpoint
318
Load program counter with program address of alternative code associated with the checkpoint
320
Execute the alternative segment of object code
322
Continue with execution of alternative segments of object code
No
322
Retry first set of code?
No
324
Exit program with error

FIG. 4

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

ATTORNEY DOCKET NO. 10001161-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

Method and Apparatus for Resuming Execution of a Failed Computer Program

the specification of which is attached hereto unless the following box is checked:

( ) was filed on _____ as US Application Serial No. or PCT International Application Number _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U S C 119
			YES: ___ NO X
			YES: ___ NO X

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE

U. S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS (patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Customer Number 022879

Place Customer Number Bar Code Label here

Send Correspondence to:
HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Direct Telephone Calls To:

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Michael L. Ziegler **Citizenship:** United States

Residence: 1189 Audrey Avenue, Campbell, California 95008

Post Office Address: 1189 Audrey Avenue, Campbell, California 95008

Inventor's Signature _____ **Date** 11/13/00

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION (continued)

ATTORNEY DOCKET NO. 10001161-1

Full Name of # 2 joint inventor: Carol L. Thompson Citizenship: United States

Residence: 6937 Calabazas Creek Circle, San Jose, California 95129

Post Office Address: 6937 Calabazas Creek Circle, San Jose, California 95129

Carol L Thompson — Nov 16, 2000

Inventor's Signature — Date

Full Name of # 3 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature — Date

Full Name of # 4 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature — Date

Full Name of # 5 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature — Date

Full Name of # 6 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature — Date

Full Name of # 7 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature — Date

Full Name of # 8 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature — Date